C Programming, Exercises for the first week

1. Write a program which prints your name on the screen. Use the NetBeans programming environment.

2. Write a program which prints the values of three variables. The types of the variables are char, int, and float. Test with different values.

3. Write a program which prints different decimal numbers. Change the precision of the outputs. Also, use output fields of different sizes.

4. Write a program that reads a forename, surname and year of birth. Use the scanf function. Then the program prints the names and the year one after another sequentially.

5. As the previous program, but now the program reads several names one after another. Design how to inform the program that there are no more names.

6. C is not a strongly typed language. To clarify this, consider a C program that contains the following declarations:

   ```c
   int i,j;
   long ix;
   short s;
   float x;
   double dx;
   char c;
   unsigned long ux;
   ```

   Determine the data type of each of the following expressions.

   a) $i + c$;
   b) $x + c$;
   c) $dx + x$;
   d) $((int)dx) + ix$;
   e) $i + x$;
   f) $s + j$;
   g) $ix + j$;
   h) $s + c$;
   i) $ix + c$;
   j) $ix + ux$;

   Make a program which tests the values of the expressions. Use different values for variables.
7. Write a function that takes an integer as a parameter and returns its absolute value.

8. Write a function that return the greatest common divisor of two integers. Use the Euclidean algorithm:

   Parameters: Two positive integers, a and b.
   Returns: The greatest common divisor, g, of a and b.
   Method:
   1. If a<b, exchange a and b.
   2. Divide a by b and get the remainder, r.
      If r=0, report b as the GCD of a and b.
   3. Replace a by b and replace b by r.
      Return to the previous step.

   Make both a recursive and iterative function. Are there differences in performances? Use so large numbers as possible.

9. Make a function fr-add(f1, f2) which returns the sum of fractions f1 and f2. Simplify the result using the Euclidean algorithm implemented in the previous exercise. Use the type struct to represent a fraction. The struct has two integer fields, nominator and denominator.

10. Make a function fr-sub(f1, f2) which returns the subtraction of f2 from f1. Simplify the result.

11. Make a function fr-mul(f1, f2) which returns the multiplication of f1 and f2. Simplify the result.

12. Make a function fr-div(f1, f2) which returns the division of f1 by f2. Simplify the result.

13. Write the following program, compile and run it. What happens? What is the problem? Does the system indicate what’s wrong? What kinds of hints does a debugging environment provide? Correct the program.

    #include <stdlib.h>
    #include <stdio.h>

    int main(int argc, char* argv[])
    {
        for (int i = 0; i <= argc; ++i)
        {
            printf("%s\n", argv[i]);
        }
        exit(EXIT_SUCCESS);
    }